A Mathematical Theory of Prolog(Lambda Calculus and Computer Science Theory)

AUTHOR(S):

Oyagi, Shigeo

CITATION:

Oyagi, Shigeo. A Mathematical Theory of Prolog(Lambda Calculus and Computer Science Theory). 数理解析研究所講究録 1984, 515: 16-61

ISSUE DATE:

1984-03

URL:

http://hdl.handle.net/2433/98378

数理解析研究所講究録
515巻 1984年 16-61

A Mathematical Theory of Prolog

Shigeo Oyagi

(Electrotechnical Laboratory)

Abstract

We present a reflexive domain E, which is very similar to the Plotkin's T^{ω}. Flat subspace E_0 of E is selected which has a regular open topology. Prolog procedure is shown to be continuous on the open subset of E_0. It is also shown that any function which is continuous on the open subset of E_0 can be extended to a continuous function on E. So Prolog procedure is continuous on E and semantics of Prolog can be defined on E.

Finally it is shown that functions of E_0 which can be approximated by step functions coincides with the continuous function of E on $E_0 \setminus A$ where A is a nowhere dense subset of E_0.

Introduction

This paper provides a mathematical theory for "Prolog". Semantics of Prolog has logical, denotational and interpretive aspects. Whole aspects can be derived from one mathematical object is desierable.

The logical semantics of Prolog can be obtained from Herbrand model. However it is not sufficient if programmer use cut primitive. Cut primitive is related to both logical and procedural semantics. It reduces the search space of proof sequences and directs how to short-cut the search.

Kowalski-Emden approach to the procedural semantics of Prolog cannot treat cut-operator. Johnes-Mycroft approach is not logical.

So no one is known which succeeds to give both logical and procedural semantics of Prolog.

Prolog procedure gets a term and produce another term. A term denotes the set of constant terms. So Prolog is a function from the sets of constant terms to the sets of constant terms.

Let E_0 be the power set of the constant terms. Prolog is shown to be a continuous function on the open subset of E_0 which has regular open topology. From the logical point of

view, Prolog procedure is identified with its invariant set. So it is comparable with invariant set of or-parallel evaluator of Horn sentence.

To assign program construct a higher order function, a reflexive domain is required.

This is a denotational way of describing the semantics of programing language.

The space E which is very similar to the Plotkin's $T^{(\omega)}$ is introduced. The element of E is a pair $\langle x,y\rangle$ where x and y are sets of constant terms satisfying $x \cap y = \not\emptyset$. The space E is shown to be reflexive.

E_0 can be embedded into E by identifying x and $\langle x,x^* \rangle$ where x^* is the largest element of $\{y \mid y \cap x = \not\emptyset\}$.

The key point is that a continuous function on the open subset of E_0 can be extended to a continuous function on E.

Prolog procedure is a continuous function on the open subset of E_0. So it can be extended to a continuous function on E.

1. Prolog procedure on the space E_0

In this section the space E_0 is described and Prolog procedure is defined as a function on this space. It is proved that Prolog procedure is defined and continuous on

the open subset of E_0 .

The set of finite trees, FT, is defined inductively as follows.

[Definition 1]

FT is a minimal set satisfying the following conditions.

(i) $c_n \in FT$ where $n \geq 1$

(ii) If $\alpha_1, \cdots, \alpha_m \in FT$ then

$$f_m^n(\alpha_1, \cdots, \alpha_m) \in FT$$

for $n,m : 1 \leq n,m \leq N$

Here c_n is a constant symbol

and f_m^n is a function symbol

FT(N) is the set of elements of FT in which only constants with index not larger than N appear.

[Definition 2]

A finite tree $F \in FT$ which has constants $c_{n_1}, \cdots, c_{n_r}$ $(n_1, \cdots, n_r > N)$ as subexpressions is denoted as

$$F = F(c_{n_1}, \cdots, c_{n_r}).$$

Define function b and its extension $\bar{b}$ as

$$b : FT \rightarrow P(FT(N))$$

$$; \; b(F(c_{n_1},\cdots,c_{n_r})) = \{F(x_1,\cdots,x_r) \mid x_1,\cdots,x_r \in FT(N)\}$$

$$\bar{b} : P(FT) \to P(FT(N))$$

$$; \; \bar{b}(x) = \bigcup_{F \in x} b(F)$$

Let α and β be subsets of FT then above(α) and under(β) is defined as

$$\text{above}(\alpha) = \{ x \mid x \supset \alpha \}$$

$$\text{under}(\beta) = \{ x \mid x \subset \beta \}$$

respectively.

Finally define $\tilde{\Sigma}_0$ as

$$\tilde{\Sigma}_0 = \{ \text{above}(\alpha) \cap \text{under}(\beta^c)$$

α is a finite subset of FT(N)

and there exists a finite subset β' of FT such that

$$\beta = \bar{b}(\beta')$$

E_0 is a sapce P(FT(N)) with the topology generated by $\tilde{\Sigma}_0$.

Example 1

Define function fail(p(f(x)),z) as

$$(\text{fail}(p(f(x)),z))(u) = \begin{cases} z & \text{in case } u \subset \{p(f(x)) \mid x \in FT(N)\}^c \\ \emptyset & \text{else} \end{cases}$$

then fail(p(f(x)),z) is continuous.

Example 2

Define function $\|q(x,f(x),y) \Leftarrow q(f(x),x,y)\|$

as

$$\|q(x,f(x),y) \Leftarrow q(f(x),x,y)\|\ (u) = \{q(f(x),x,y) \mid q(x,f(x),y) \in u\}.$$

Then $\|q(x,f(x),y) \Leftarrow q(f(x),x,y)\|$ is continuous.

Here we define some functions to construct Prolog procedure.

[Definition 3]

Let $\{f_n \mid n \in I\}$ be a set of partial functions on the space E_0 and $(I,\prec)$ be a directed set. Then finite limit of $\{f_n \mid n \in I\}$ is defined as follows.

$$\text{F-lim } f_n(x) = \begin{cases} f_N(x) \text{ in case there exists } N \in I \text{ such that} \\ \qquad f_n(x) \text{ is defined and} \\ \qquad f_n(x) = f_N(x) \text{ for } n \succ N \\ \text{undefined otherwise} \end{cases}$$

[Definition 4]

If $F_i(x_1,\ldots,x_n,u)$ is a partial function of u on the space E_0 for arbitrary partial functions $x_1,\ldots,x_n$ then the notation

$$\begin{cases} x_1(u)=F_1(x_1,\ldots,x_n,u) \\ \quad\vdots \\ x_n(u)=F_n(x_1,\ldots,x_n,u) \end{cases}$$

denotes functions

$$x_i(u)=F\text{-}\lim F_i([x_1]_{\alpha_1},\ldots,[x_n]_{\alpha_n},u)$$

for $i=1,\ldots,n$

where $[x_1]_{\alpha_1},\ldots,[x_n]_{\alpha_n}$ are defined as follows.

$$\begin{cases} [x_1]_{F_1(\alpha_1,\cdots,\alpha_n)}(u) = F_1([x_1]_{\alpha_1},\ldots,[x_n]_{\alpha_n},u) \\ \quad\vdots \\ [x_n]_{F_n(\alpha_1,\cdots\alpha_n)}(u) = F_n([x_1]_{\alpha_1},\ldots,[x_n]_{\alpha_n},u) \end{cases}$$

and $[x_1]_0,\ldots,[x_n]_0$ are totally undefined functions.

[Definition 5]

While sentence is defined as follows.

$$\begin{aligned} &(\text{While Exp do S end})(x) \\ =&\begin{cases} (\text{While Exp do S end})(S(x))\cup x \\ \qquad\qquad \text{in case } \mathrm{Exp}(x)\supset\{0,1\} \\ (\text{While Exp do S end})(S(x)) \end{cases} \end{aligned}$$

$$\begin{cases} & \text{in case } \mathrm{Exp}(x) \supset \{1\} \text{ and } \mathrm{Exp}(x) \not\supset \{0\} \\ x & \text{in case } \mathrm{Exp}(x) \supset \{0\} \text{ and } \mathrm{Exp}(x) \not\supset \{1\} \\ \text{undefined} & \text{otherwise} \end{cases}$$

[Definition 6]

Case construct is defined as follows.

$$(\text{Case } x \text{ of } y_1 : z_1 ; \ldots y_n : z_n ; \text{end})(u)$$

$$= \bigcup_{y_i \in x(u)} z_i(u)$$

where x and z_i i=1,...,n are functions on E_0.

[Definition 7]

If z_i are functions for i=1,...,n then

$$z_1 ; \ldots z_{n-1} ; z_n = z_n \circ \ldots \circ z_2 \circ z_1$$

and

begin S end = S

[Definition 8]

Natural numbers are defined as follows.

$$\begin{cases} 0 = c_1 \\ 1 = f_1^1(0) \\ n = f_1^1(n-1) \end{cases}$$

The set of natural numbers are denoted as Nat.

[Proposition 1]

A partial function f on FT(N) can be extended to a continuous function $\bar{f}$ on P(FT(N)) as

$$\bar{f}(z_1, \ldots, z_n) = \cup \{ f(x_1, \ldots, x_n) \mid x_i \in z_i \text{ for } i=1,\ldots,n \}$$

The notation f shall be used instead of $\bar{f}$.

[Definition 9]

Pair function $\langle x,y \rangle$ and assignment $u \leftarrow v$ are defined recursively as follows.

$$[x,y] = f_2^1(x,y)$$

$$\langle\ \rangle = [0,0],\ \langle x_1 \rangle = [x_1, \langle\ \rangle]$$

$$\langle x_1, \ldots, x_n \rangle = [x_1, \langle x_2, \ldots, x_n \rangle]$$

$$\text{length}(\langle x_1, \ldots, x_n \rangle) = n$$

$$\text{Push}(a, \langle x_1, \ldots, x_n \rangle) = \langle a, x_1, \ldots, x_n \rangle$$

$$\text{Pop}(\langle x_1, \ldots, x_n \rangle) = \langle x_2, \ldots, x_n \rangle$$

$$i\uparrow(\langle x_1, \ldots, x_n \rangle) = x_i$$

$$(i \leftarrow v)(\langle x_1, \ldots, x_n \rangle) = \langle x_1, \ldots, x_{i-1}, v, x_{i+1}, \ldots, x_n \rangle$$

$$(i.u \leftarrow v)(z) = (i \leftarrow (u \leftarrow v)(z_i))(z)$$

$$(i.u)(z) = u(i(z))$$

[Definition 10]

"if x then y else z fi"

means if x=1 then y and if x=1 then z.

"if x then y"

is an abrieviation of "if x then y else fi"

[Definition 11]

"let x = v in F(x)"

$\equiv$ F(v)

"F(x) where x = v"

$\equiv$ F(v)

Pure Prolog with no assert statement and evaluable predicate can be defined syntactically as follows.

Program ::= Clause-groups

Clause-groups ::= Clause-group|
Clause-group Clause-groups

Clause-group ::= Clauses<i,j>

Clauses<i,j>::=Clause<i,j>|
Clause<i,j> Clauses<i,j>

Clause<i,j> ::= Clause-Head<i,j> :- And-part

And-part ::= terms|Λ|!|terms| terms!

terms ::=term|terms|terms|terms,terms

term ::= $c_1|\ldots|c_N|f_1^1(arg)|\ldots|f_N^N(arg,\ldots,arg)$

arg ::= term|$x_1|x_2|\cdots|x_n|\cdots$

Clause-Head<i,j> ::= $f_i^j(arg,\ldots,arg)$

Prolog program is a list of Clause-groups. A clause-

group is a list of Clauses which have heads that starts from the same predicate symbol.

A Clause is a list of the form

"Head:- a list of terms"

and the list of terms has cut symbol ! or, as delimitters.

If a program has a clause-group

$$d_1\ d_2 \cdots d_n$$

we can specify the next clause. This function we denote by next(d).

$$next(d_i) = d_{i+1} \quad \text{for } i = 1,\ldots,n-1$$

$$next(d_n) = \text{undef}$$

The head(d) is a Clause-Head part and the tail(d) is an And-part for Clause d.

The pred-name(d) is the name of the clause group of d.

The first(<i,j>) denotes the first clause of the Caluse group<i,j>.

Let d be a clause

$$d = \text{"p:-}\ \varepsilon_1 p_1 \cdots \varepsilon_n p_n\ \varepsilon_{n+1}\text{"}$$

where ε_i = ! or ,

The term p_i is specified by term(d,i)

We define functions Cal, Ret, Fail, Suc, Unif on P(FT(N)) as follows.

$$Cal(d,i) = \| p \Leftarrow p_i \|$$

$$Ret(d,i) = \| p_i \Leftarrow p \|$$

$$Fail(d) = fail(p)(l)$$

$$Suc(d) = \| p \Leftarrow l \|$$

$$Unif(d) = \| p \Leftarrow p \|$$

[Proposition 2]

Cal, Ret, Fail, Suc, Unif are continuous functions.

For Prolog program we shall define a function on E_0 which we call Abstract Prolog. First we will consider the data structures which shall be handled by the Abstract Prolog.

stack : list-of stack-element

stack-element : ⟨ clause-name, stack-pointer-of caller, logical-state, subprocess-status ⟩

subprocess-status : list of ⟨clause-name, stack-pointer, logical-state⟩

state: ⟨local-stack, mode, stack⟩

local-stack : list of local-state

local-state : list of variable-assigiment

variable-assignment :

⟨stp, stpl,

We assign integers to the selecters.

clause-name = 1

stack-pointer-of-caller = 2

logical-state = 3

subprocess-status = 4

stack-pointer = 2

logical-state = 3

local-stack = 1

mode = 2

stack = 3

If u has a data type of state then

(stack.clause-name←d)

replaces the clause-name slot of the stack part of u by d if it is applied to u.

Let clause group <i,j> be given. The process of its execution we call "process". The process status is composed of the clause name it is currently executing, the position of the process-status in the stack and the state descriptions of the subprocess which it should call.

Subprocess status is made of three components. First one is the next alternative clause of the clause under execution.

The second is the base point of data for the process in the stack.

The third one retains the partial return value of the process.

If input Prolog sentence d_0 is ":-p_0" then the corresponding element u_0 of E_0 is Cal(d_0)(1).

The element u_0 is an input for the Abstract Prolog.

We define the function pred-name′ as

$$\text{pred-name}'(u) = \bigcup_{1 \le i,j \le N} \| f_i^j(x_1, \ldots, x_i) \Leftarrow \langle i,j \rangle \| \ (u)$$

$$\cup \bigcup_{1 \le i \le N} \| c_i \Leftarrow \langle i \rangle \| \ (u)$$

The initial state of the stack becomes

$$\bigcup_{i,j} \langle\langle \text{pred-name}'(u_{ij}), 1, u_i, \langle \text{first}(\text{pred-name}(u_{ij}), \text{undef}, \text{undef} \rangle\rangle\rangle$$

where

$$u_{ij} = \begin{cases} \| f_i^j(x_1, \ldots, x_i) \Leftarrow f_i^j(x_1, \ldots, x_i) \| \ (u) \\ \qquad \text{for } i,j = 1, \ldots, N \\ \| c_j \Leftarrow c_j \| \ (u) \text{ for } i = 0 \text{ and } j = 1, \ldots, N \end{cases}$$

It is depicted in the figure 1.

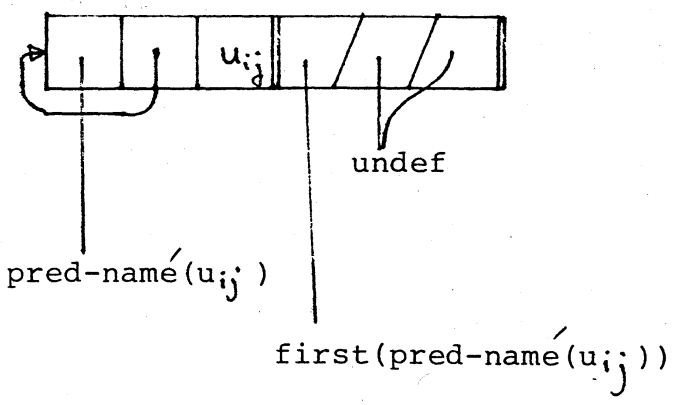


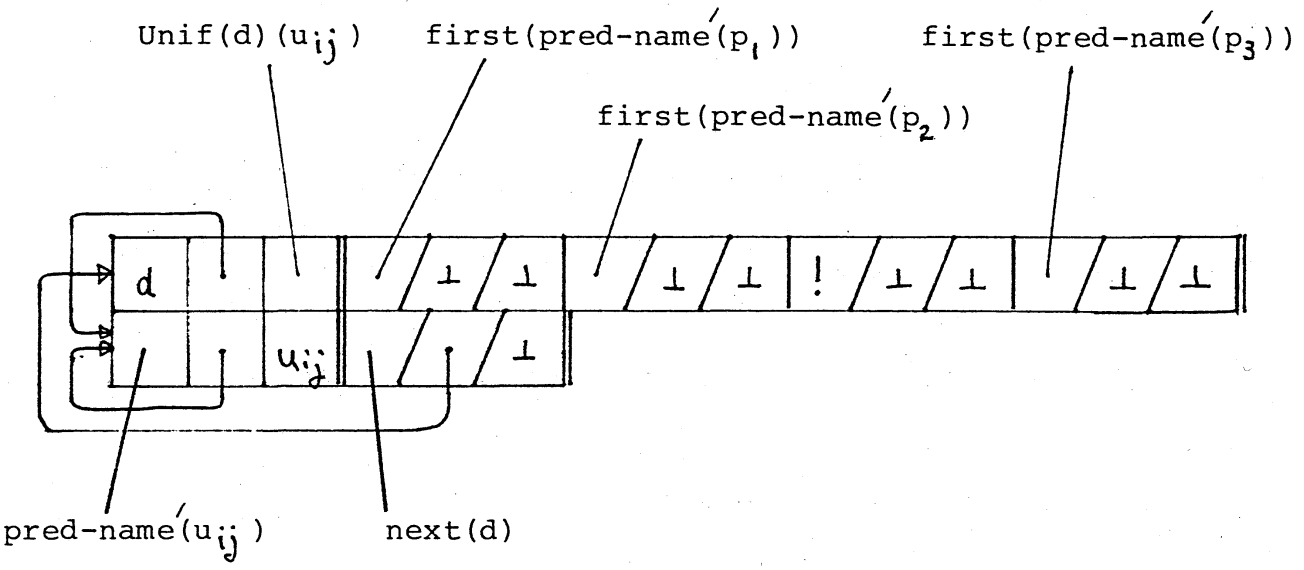
fig. 1

In case

$$\text{first}(\text{pred-name}'(u_{ij})) = d = \text{"p:-}p_1, p_2 ! p_3\text{"}$$

and $\quad \text{Suc}(d)(u_{ij}) \ni 1$

the stack becomes as in the figure 2.



⊥=undef

fig. 2

A procedure which define "process" has four parameters. First one is the kind of process. A "process" has two kinds. One is clause selection process usually called "or-process". The other is subprocess monitoring process called "and-process". They are further descriminated by clause-group names and clause names respectivly.

Second parameter is the stack base position for the process. This is specified by pt. The third one is the position of the current term in the clause description.

The last parameter is the mode of execution,"normal" order or "reverse" order.

Procedures require "local-stack" for local variables and a mode parameter "mode".

The local stack should not be confused by the local stack of the usual Prolog interpreter.

We introduce here the procedure declaration.

[Definition 12]

"Procedure m(x_1,...,x_n)
dcl a_1,...,a_m local

Body

end"

$\equiv$ $m(x_1,\ldots,x_n)$

=local-stack←Push(⟨undef,...,undef⟩,local-stack);

Body

local-stack ← Pop(local-stack)

where a_1=local-stack.1.1

a_m=local-stack.1.m

Abstract Prolog is defined as follows.

$init_{ij}(u)=\langle loc,undef,s_{ij}\rangle$

where loc= ⟨ ⟩

$s_{ij}=\langle\langle pred\text{-}name'(u_{ij}),1,u_{ij},$

$\langle first(\langle i,j\rangle),undef,undef\rangle\rangle\rangle$

$$u_{ij}=\begin{cases} \| f_i^{\phi}(x_1,\ldots,x_i)\Leftarrow f_i^{\phi}(x_1,\ldots,x_i)\| \ (u) \\ \qquad\qquad \text{for } i,j=1,\ldots,N \\ \| c_j \Leftarrow c_j \| \ (u) \\ \qquad\qquad \text{for } i=0,j=1,\ldots,N \end{cases}$$

```
Main(w)
      = ∪ [if k_ij (stack.1.logical-state(w) then
        i,j     stack.1.logical-state(
                  process(⟨i,j⟩1,1,norm)(w))]
```

where

$$k_{ij} = \begin{cases} (f_i^j(x_1,\ldots,x_i) \Leftarrow f_i^j(x_1,\ldots,x_i)) \\ \qquad \text{for } i,j=1,\ldots,N \\ (c_j \Leftarrow c_j) \\ \qquad \text{for } i=0,j=1,\ldots,N \end{cases}$$

```
procedure process(q,pt,ix,norm)
      dcl d local
      d←stack.pt.subprocess-status.ix.clause-name;
      if d≠undef then
         begin
           stack.pt.subprocess-status.ix.clause-name←next(d↑);
           if Suc(d↑)(stack.pt.logical-state↑) then
              begin
                process(d↑,pt,ix,norm);
                if mode=fail then
                process(q,pt,ix,norm)
              end {begin};
```

```
            if Fail(d↑)(stack.pt.logical-state↑) then
               process(q,pt,ix,norm)

         end {begin}
      end {procedure}


procedure process(q,pt,ix,reverse)
      dcl stpl,d,r,dl local
      stpl←stack.pt.subprocess-status.ix.stack-pointer↑;
      d←stack.stpl↑.clause-name↑;
      r←length(d↑);
      if r≠0 then
         process(d↑,stpl↑,undef,reverse);
      if mode=fail then
         begin
           dl←stack.pt.subprocess-status.ix.clause-name↑;
           stack.pt.subprocess-status.ix.clause-name←next(dl↑);
           While dl↑≠undef and mode↑=fail
             do
               stack←Pop(stack↑);
               process(pred-name(dl↑),pt,ix,norm)
             end {while}
         end {begin}
      end {procedure}

procedure process(d,pt,ix,t)
```

```
= dcl stp,ixl,r,dl,ℓ,stpl local
 Case t of
 norm:
   stack←Push(⟨d,pt,v,⟨⟨first(pred-name′(p1)),
     undef, undef⟩,···,⟨first(pred-name′(pr))⟩⟩⟩,
     stack)
       where v = Unif(d)∘Cal(stack.pt.clause-
           name↑,ix)(w) w=stack.pt.subproccess-status.
             (ix-1).logical-state↑ in case ix≠1,
             stack.pt.logical-state↑ in case ix=1
           pi=term(d,i)  for i = 1,...,rl,
           rl=length(tail(d)) ;
   stack.pt.subprocess-status.ix.stack-pointer
         ←length(stack↑);
   if length(tail(d)) = 0 then
     begin
       mode←suc;
       stack.pt.subprocess-status.ix.logical-state
        ←Ret(stack.pt.clause-name↑,ix)(
            stack.length(stack↑).
                       logical-state↑)
     end {begin}
   else
     begin
       stp←length(stack↑);
       ixl←1;
```

```
                r←length(d);
                mode←suc;
                while-sentence
            fi
        reverse:
            ixl←length(tail(d));
            if ixl↑=0 then
              begin
                mode←fail;
                stack←Pop(stack↑)
              end {begin}
            else
              begin
                mode←fail;
                stp ←length(stack↑);
                r←ixl↑;
                while-sentence
        end {Case}
where while-sentence=
      While  l≤ixl↑≤r↑
        do
          dl←stack.stp.subprocess-status.
              ixl.clause-name↑;
          if dl↑= !  and mode↑=fail then
            begin
```

```
    ℓ ← length(stack↑);
    While  stp↑ ≤ ℓ↑
    do
      stack←Pop(stack↑);
      ℓ ← ℓ↑ - 1
     end {while};
     stack←PoP(stack↑);
     stack.pt.subprocess-status.ix.
     clause-name←first(head(d));
     ix1←n↑+2
   end {begin}
else
   begin
     if d1↑=!  and mode↑= suc then
       begin
         if ix1↑= 1 then
           begin
             stack.stp1.subprocess-status.
             ix1↑.logical-state←stack.
             stp.logical-state↑;
             ix1←ix1↑+1
           end {begin}
         else
           begin
             stack.stp1↑.subprocess-status.
             ix1↑.logical-state
             ←stack.stp1↑.subprocess-status.
```

```
                    (ix1↑-1).logical-state↑;
                    ix1←ix1↑+1
                  end {begin}
                fi
              end {begin}
            else
              if mode↑= suc then
                  process(pred-name(d1↑),stp1↑,ix1↑,norm)
              else
                process(pred-name(d1↑),stp1↑,ix1↑,reverse)
              fi;
              Case mode↑of
                suc:
                 ix1←ix1↑+1
                fail:
                 ix1←ix1↑-1
              end {Case}
            fi
          end {begin}
        fi
      end {while}
```

```
        Case ixl↑of
          0: mode←fail;
             stack←Pop(stack↑);
             stack.pt.subprocess-status.ix.
             clause-name←first(head(d))
          r↑1: mode←suc;
             stack.pt.subprocess-status.ix.
             logical-state←Ret(d,ix)(stack.
             stp .subprocess-status.r.
             logical-state↑
          r↑2:
        end {Case}
      end {procedure}
```

Some properties of Prolog procedure shall be stated here.

[Lemma 1]

Let f be a partial function on E_0.
If for any $x \in D(f)$ there exists an open neighborhood U of x such that f is continuous on U then D(f) is open and f is continuous on D(f).

[Lemma 2]

Let $\alpha, \beta, \gamma, \delta$ are totaly defined continuous functions on E_0 and $Im(\alpha)=Im(\beta)=\{\{1\},\phi\}$.
If $\alpha(x) \cap \beta(x) \neq 1$ then the function defined as

$$f(x)=\text{if } \alpha(x) \text{ then } f(\gamma(x)); \text{ if } \beta(x) \text{ then } \delta(x)$$

has an open domain and it is continuous.

(Proof)

Let x be an element of D(f).

Define U_β and U_α as

$$U_\beta=\{u \mid \beta(u)\ni 1\}$$

and

$$U_\alpha=\{u \mid \alpha(u)\ni 1\}.$$

From $x\in D(f)$ $x\in U_\alpha$ or $x\in U_\beta$.

In case $x\in U_\beta$, f(u) is defined for $u\in U_\beta$ and f coincides with a continuous function δ on U_β.

f is continuous on U_β.

If $x\in U_\alpha$ then there exists N such that

$$\gamma^N(x)\in U_\beta, \quad \gamma^{N-i}(x)\in U_\alpha \text{ for } i=1,\ldots,N$$

and $f(x)=\delta(\gamma^N(x))$

by the definition of recursion.

Define V as

$$V=U_\alpha\cap(\gamma^N)^{-1}(U_\beta)\cap\bigcap_{1\le i\le N}(\gamma^{N-i})^{-1}(U_\alpha)$$

then V is an open neighborhood of x and

$$f(v)=\zeta(\gamma^N(v)) \text{ for } v\in V.$$

So f is continuous on V.

From Lemma 1 and the above fact, the Lemma 2 holds.

q.e.d.

Extended version of Lemma 2 can be easily obtained and if we apply it to the Abstruct Prolog then the following proposition will be shown.

[Proposition 3]

Abstruct Prolog has an open domain and it is continuous.

2. Reflexive domain E

In this section a space E is introduced. It shall be shown that E is a continuous lattice, E is reflexive and the paradoxical combinator coincides with the least fixed point operator. This section is a preparation for the next section in which the relationship between E and E_0 shall be

discussed.

[Definition 13]

Let above'(α) be the set

$$\text{above}'(\alpha)=\{x \mid x \supset \alpha, x \subset FT\}$$

for a finite subset α of FT.

We introduce the topology $\mathcal{I}'$ into P(FT) which is generated by the sets

$$\mathcal{I}_0'=\{\text{above}'(\alpha) \mid \alpha \text{ is finite}\}.$$

[Definition 14]

In case a finite tree F has constant symbols $c_{n_1},\ldots,c_{n_r}$ for $n_1,\ldots,n_r>N$ on its leaves, F is denoted as

$$F=F(c_{n_1},\ldots,c_{n_r}).$$

Define function b' : FT$\to$P(FT) as

$$b'(F(c_{n_1},\ldots,c_{n_r}))$$

$$=\{F(x_1,\ldots,x_r) \mid x_1,\ldots,x_r \in FT\}.$$

Further the extention $\bar{b}'$ of b' is defined as

$$\bar{b}'(x) = \bigcup_{F \in x} b'(F)$$

[Proposition 4]

$\bar{b}'$ is a continuous retraction map.

[Proposition 5]

The map r defined as

$$r : P(FT(N)) \times \bar{b}'(P(FT)) \to P(FT(N)) \times \bar{b}'(P(FT))$$

;

$$r(\langle x,y \rangle) = \begin{cases} \langle x,y \rangle & \text{when } x \cap y = \not\emptyset \\ \top & \text{else} \end{cases}$$

is a retraction map.

[Definition 15]

Define space E as

$$E = r(P(FT(N)) \times \bar{b}'(P(FT))).$$

The first element of the pair $\langle x,y \rangle$ of E represents positive information and the second element represents

negative one.

[Definition 16]

Let β and α be as

$$\beta=\langle F, b'(G)\rangle,\ \alpha=\langle \bigcup_{n=1}^{l} F_n,\ \bigcup_{n=1}^{m} b'(G_n)\rangle$$

for $F, F_n \in FT(N)$ $G, G_n \in FT$

A pair function $g(\beta,\alpha)$ for such pair (β,α)s is defined as follows.

$$g(\beta,\alpha) =$$
$$\langle \not\varnothing, \langle F, b'(G), d_1(F_1), \ldots, d_l(F_l), e_1(b'(G_1)), \ldots, e_m(b'(G_m))\rangle\rangle$$

where $d_i=(f_i^3)^{2i}$, $e_i=(f_i^3)^{2i+1}$ for
$i=1,\ldots,\max(l,m)$

To assure the uniqueness of the function values, the order of arrangement of F_i, G_i must be determined.

Let $n(f_i^{\beta}(\gamma_1,\ldots,\gamma_i))$ be as

$$n(f_i^{\beta}(\gamma_1,\ldots,\gamma_i))=n(f_i^{\beta})^{P_r(1)^{n(\gamma_1)}\cdots P_r(i)^{n(\gamma_i)}}$$

where $n(f_i^j)=Pr(\frac{1}{2}(i+j)(i+j-1)-i+1)$.

and Pr(n) denotes a n-th prime number

Then the order is defined as

$$n(F_1)\leq\ldots\leq n(F_\ell)$$

and $$n(G_1)\leq\ldots\leq n(G_m)$$

Using this pair function g the graph of a continuous function of E can be defined as in the following.

[Definition 17]

Let $\beta=\langle F,b'(G)\rangle$ and $\alpha=\bigsqcup_{i=1}^{\ell}\langle F_i,b'(G_i)\rangle$.

Define the function τ_α^β as follows.

$$\tau_\alpha^\beta(x)=\begin{cases}\beta & \text{if } \alpha\sqsubseteq x\\ \\ \bot & \text{else}\end{cases}$$

Then the graph of arbitrary continuous function f is defined as

$$\lambda x f(x)=\bigsqcup_{\tau_\alpha^\beta\sqsubseteq f} g(\beta,\alpha)$$

[Proposition 6]

Define function γ as

$$\gamma(u)=\bigsqcup_{\lambda z\, \tau_\alpha^\beta(z)\sqsubseteq\gamma} \tau_\alpha^\beta(u)$$

for any $u\in E$.

Then

1) γ is continuous.

2) $[\lambda z f(z)](u)=f(u)$

To assure the soundness of this graph interpretation, we will show the fixed point theorem.

[Proposition 7]

The paradoxical combinator Y defined as

$$Y=\lambda u\lambda x\ u(x(x))(\lambda x u(x(x)))$$

coincides with the least fixed point operator fix defined as

$$fix(x)=\bigsqcup_n f^n(x)$$

in the graph model of E.

(Proof)

Let N be a function on $D=\{b'(G) \mid G \in FT\}$ defined as

$N(b'(G))$=the number of nodes in the tree G.

Then trivialy

$$N(g(\langle F,G\rangle,\langle \cup F_n, \cup G_n\rangle)) > N(F_\eta),\ N(G_\eta)$$

holds.

From Scott[4], this is a sufficient condition for the theorem.

q.e.d.

3. The relationship between E_0 and E

In this section how E_0 can be embedded into E shall be clarified and the following properties of E_0 shall be shown.

(i) Any continuous function on the open subset of E_0 can be extended to a continuous function on the space E.

(ii) Any lower semi-continuous function on the space E_0 can be extended to a continiuous function of E

except for some nowhere dense subset.

From the fact (i) we can conclude that Abstract Prolog can be extended to a continuous function on E.

[Definition 18]

Define E_0^* as

$$E_0^* = \{ x \mid x \text{ is maximal in } E \setminus \{T\} \}$$

[Proposition 8]

The map $I : E_0^* \to E_0 , \langle x,y \rangle \mapsto x$

is homeomorphic

(Proof)

I is clearly one to one.

$\langle x,y \rangle \in \text{above}(\alpha) \cap \text{above}(\beta)$

$\Leftrightarrow x \in \text{above}(\alpha)$ and $y \supset \beta$

So

$$y \supset \beta \Rightarrow y \cap FT(N) \supset \beta \cap FT(N)$$
$$\Rightarrow x \cap \beta \cap FT(N) = \emptyset$$
$$\Rightarrow x \subset (\beta \cap FT(N))^c$$

That is $\langle x,y \rangle \in \text{above}(\alpha) \cap \text{above}(\beta)$

implies

$$x \in \text{above}(\alpha) \cap \text{under}((\beta \cap FT(N))^c).$$

Conversely if $x \in \text{above}(\alpha) \cap \text{under}((\beta \cap FT(N))^c)$

then

$$x \cap \beta \cap FT(N) = \emptyset.$$

Trivially

$$x \cap (\beta \setminus FT(N)) = \emptyset$$

So

$$x \cap \beta = \emptyset.$$

If $\langle x, y \rangle \in E_0$ then

$$y \supset \beta$$

because y is a maximal element satisfying

$$x \cap y = \emptyset.$$

This shows

$$\langle x, y \rangle \in \text{above}(\alpha) \cap \text{above}(\beta).$$

q.e.d.

We can identify the space E_0 with the space E_0^*.

[Definition 19]

The function $\chi^{\gamma}_{\alpha,\beta}$ is defined as

$$\chi^{\gamma}_{\alpha,\beta}(x) = \begin{cases} \gamma & \text{in case } x \supset \alpha \text{ and } x \cap \beta = \emptyset \\ \emptyset & \text{else.} \end{cases}$$

where γ, α are finite sets of FT(N)

and β is a set described as

$$\beta = \overline{b}(\beta')$$

for some finite subset β' of FT.

[Proposition 9]

$\chi^{\mathcal{F}}_{\alpha,\beta}$ is a continuous function on E_0.

(Proof)

$$[\alpha, \beta^c] = \{x \mid x \supset \alpha \text{ and } x \subset \beta^c\}$$

is both open and closed.

So this is trivial.

q.e.d.

[Proposition 10]

$x \cup y$ is continuous on E_0

(Proof)

It is possible to select finite sets α_1, α_2 such that

$$\alpha = \alpha_1 \cup \alpha_2 \text{ and } x \supset \alpha_1, y \supset \alpha_2 .$$

Trivially

$$x \in [\alpha_1, \beta^c], y \in [\alpha_2, \beta^c]$$

hold and

$$u \cup v \in [\alpha, \beta^c]$$

for any $u \in [\alpha_1, \beta^c]$ and any $v \in [\alpha_2, \beta^c]$

q.e.d.

[Proposition 11]

If f_n is continuous on E_0 for $n \geq 1$

then the function

$$\cup_{n \geq 1} f_n$$

is lower semi-continuous.

(Proof)

Assume

$$(\cup f_n)(x) \in [\alpha, FT(N)]$$

for open set $[\alpha, FT(N)]$.

Trivially there exists a number N

such that.

$$\cup_{n \leq N} f_n(x) \in [\alpha, FT(N)]$$

From the continuity of $\cup_{n \leq N} f_n$ there exists

a neighborhood $[\gamma, \delta^c]$ of x such that

$$\cup_{n \leq N} f_n(y) \in [\alpha, FT(N)]$$

for any $y \in [\gamma, \delta^c]$

This shows

$$(\cup f_n)([\gamma, \delta^c]) \subset [\alpha, FT(N)]$$

q.e.d.

5.

[Proposition 12]

If f is lower semi-continuous on E_0

then

$$f = \bigcup_{\chi^{\gamma}_{\alpha,\beta} \subset f} \chi^{\gamma}_{\alpha,\beta}$$

(Proof)

Let x be an arbitrary element of E_0.

For any finite subset α_0 of f(x) there exist γ and δ such that.

$$x \in [\gamma, \delta^c]$$

and $f([\gamma, \delta^c]) \supset \alpha_0$.

So

$$f \supset \chi^{\alpha}_{\gamma,\delta}$$

It means

$$f(x) \supset \bigcup_{\chi^{\alpha}_{\gamma,\delta} \subset f} \chi^{\alpha}_{\gamma,\delta}(x) \supset \alpha_0 .$$

While f(x) can be written as

$$f(x) = \bigcup_i \alpha_i$$

for finite sets α_i with $i \geq 0$.

So

$$f(x) \supset \bigcup_{\chi^{\gamma}_{\alpha,\beta} \subset f} \chi^{\gamma}_{\alpha,\beta}(x) \supset \alpha_i = f(x)$$

q.e.d.

[Definition 20]

Define the function $\eta^{\gamma}_{\alpha,\beta}$ as

$$\eta^{\gamma}_{\alpha,\beta}(x) = \begin{cases} \gamma^c & \text{in case } x \in [\alpha, \beta^c] \\ FT(N) & \text{else.} \end{cases}$$

[Proposition 13]

$\eta^{\gamma}_{\alpha,\beta}$ is continuous.

[Proposition 14]

If f is lower semi-continuous then

the closure of the set $\{x \mid f(x) \neq \bigcap_{\eta^{\gamma}_{\alpha,\beta} \supset f} \eta^{\gamma}_{\alpha,\beta}(x)\}$

does not include any open set except ϕ.

(Proof)

Let U be the set

$$U = \{x \mid f(x) \neq \bigcap_{\eta^{\gamma}_{\alpha,\beta} \supset f} \eta^{\gamma}_{\alpha,\beta}(x)\} .$$

For any $x \in U$ we can select $c(x) \in FT(N)$

such that

$$f(x) \ni c(x) \text{ and } \bigcap_{\eta^{\gamma}_{\alpha,\beta} \supset f} \eta^{\gamma}_{\alpha,\beta}(x) \ni c(x) .$$

Assume the closure of U includes a non-null

open set.

5.

Then there must exist an open set

$[\alpha_0, \beta_0^c] \neq \emptyset$ such that

$$[\alpha_0, \beta_0^c] \subset \bar{U}$$

The function c determines a partition Δ defined as

$$\Delta = \{ C^{-1}C(x) \mid x \in U \}$$

We denote $C^{-1}C(x)$ by $U_{C(x)}$.

Then

$$[\alpha_0, \beta_0^c] \subset \bigcup_{x \in U} \bar{U}_{C(x)}$$

The set $\{ \bar{U}_{C(x)} \mid x \in U \}$ is countable. So we can denote the element by V_n.

Assume there exists index n such that V_n includes an open set.

Then there exists an open set $[\varepsilon, \delta^c]$ such that

$$V_n \supset [\varepsilon, \delta^c]$$

So there exists an element x_0 such that

$U_{c(x_0)}$ is dense in $[\xi, \delta^c]$

From the lower semi-continuity of f this implies

$$f(x) \geqslant c(x_0) \text{ in } [\xi, \delta^c]$$

So

$$\eta^{\{c(x_0)\}}_{[\xi,\delta^c]} \supset f$$

This shows

$$\eta^{\alpha}_{(\alpha,\beta)} \cap \eta^{\alpha}_{(\alpha,\beta)} \supset f \quad \eta^{\alpha}_{\alpha,\beta}(x) \geqslant c(x_0) \text{ in } [\xi, \delta^c]$$

This contradicts.

So V_n does not include any non empty open set for any n.

V_0 does not include any non empty open set. So

$$[\alpha_0, \beta_0^c] \setminus V_0 \neq \emptyset$$

and there exists non-empty open set $[\alpha_1, \beta_1^c]$ such that

$$[\alpha_1, \beta_1^c] \subset [\alpha_0, \beta_0^c] \setminus V_0$$

This procedure can be applied successively for any natural number n.

So there exists $[\alpha_n, \beta_n^c]$ for $n \geq 0$ so that

$$[\alpha_n, \beta_n^c] \subset [\alpha_{n-1}, \beta_{n-1}^c] \setminus V_{n-1}$$

are satisfied.

If

$$\bigcap_{n=0}^{\infty} [\alpha_n, \beta_n^c] = \emptyset \qquad \text{holds}$$

then

$$\bigcup_{n=0}^{\infty} \alpha_n \cap \beta_n \neq \emptyset$$

must be satisfied.

Let a be the element of $\bigcup_{n=0}^{\infty} \alpha_n \cap \beta_n$

Then there exists a number N such that

$$a \in \bigcup_{n=0}^{\infty} \alpha_n \cap \beta_n .$$

This means

$$\bigcap_{n=0}^{N} [\alpha_n, \beta_n^c] = \emptyset$$

So

$$[\alpha_N, \beta_N^c] = \emptyset$$

This contradicts.

This shows

$$\bigcap_{n=0}^{\infty} [\alpha_n, \beta_n^c] \neq \emptyset$$

However

$$\bigcap_{n=1}^{\infty} [\alpha_n, \beta_n^c] \subset \bigcap_{n=1}^{\infty} ([\alpha_{n-1}, \beta_{n-1}^c] \setminus V_{n-1})$$

$$\subset \bar{U} \setminus \bigcup_{n=0}^{\infty} V_n = \emptyset$$

This also contradicts.

q.e.d.

[Definition 21]

A continuous function ψ from E to E is defined as

$$\psi_\alpha^\beta(x) = \begin{cases} \beta & x \supset \alpha \\ \emptyset & \text{else} \end{cases}$$

Further the function $\hat{f}$ is defined as

$$\hat{f} = \bigcup_{f \supset \chi_{\alpha,\beta}^\gamma} \psi_{\langle \alpha, (\beta^c)^* \rangle}^{\langle \gamma, \emptyset \rangle} \cup \bigcup_{f \subset \eta_{\alpha,\beta}^\gamma} \psi_{\langle \alpha, (\beta^c)^* \rangle}^{\langle \emptyset, (\gamma^c)^* \rangle}$$

for a lower semi-continuous function f.

[Proposition 15]

(i) $\hat{f}$ is continuous

(ii) The closure of $\{x \mid x \in E_0$ and $\hat{f}(x) \neq f(x)\}$ includes no non-empty open set.

[Proposition 16]

If function f is defined and continuous on the open subset of E_0 then f can be extended to E as a continuous function

(Proof)

Let f' be as follows.

$$f'(x)=\begin{cases} f(x) & \text{if } x\in D(f) \\ \phi & \text{else} \end{cases}$$

The function g defined as

$$g=\bigcap_{\eta^{\gamma}_{\alpha,\beta}\supset f'} \eta^{\gamma}_{\alpha,\beta}$$

is upper semi-continuous.

Moreover

$$g=f \quad \text{on } D(f)$$

f' is lower semi-continuous because D(f) is open.

So

$$\bigcup_{\chi^{\gamma}_{\alpha,\beta}\subset f} \chi^{\gamma}_{\alpha,\beta}=f \text{ on } D(f)$$

Define H as

$$H = \bigcup_{\chi^{\gamma}_{\alpha,\beta} \subset f'} \psi^{<\gamma,\beta>}_{<\alpha,(\beta^c)^*>} \cup \bigcup_{f' < \eta^{\gamma}_{\alpha,\beta}} \psi^{<\alpha,(\gamma^c)^*>}_{<\alpha,(\beta^c)^*>}$$

H is continuous and H=f on D(f).

q.e.d.

References

1. S.Oyagi.(1982)"Fixed point semantics of logical formula" seminar report of research institute of mathematical science ,kyoto university

2. Scott,D.(1972)"Continuous lattices" Lecture Notes in Mathematics Vol.274 pp97-136 Springer Verlag

3. Scott,D.(1972)"λ-Calculus and recursion theory"prc. 3rd Scandinavian Logic Symposium pp154-193 North Holland

4. Scott,D.(1980)"Lambda calculus: Some models, some philosophy" in Kleene Symposium pp381-421 North Holland

5. Scott,D.(1980)"Related theories of the λ-Calculus" Essays on Combinatory logic, lambda calculus and Formalism pp402-448 North Holland

6. Plotkin,G.D.(1978)"T^{ω} as a universal domain" J.Computer and System

Sciences 17.2 pp209-236

7. Apt. K.R. and van Emden, M.H.
Contributions to the theory of logic programming.
Journal of the ACM 29(3):841-862, 1982.

8. Jones, N. D. and Mycroft, A.
Stepwise Development of Operational and Denotational Semantics for Prolog
27 April 1983

9. Kowalski, R.
Algorithm = Logic + Control.
Communications of the ACM 22(7), 1979.